Name

Q1 - What's a relational database?

Q2 - What are the different "table relationships" you know?

Q3 - Consider this e-library service. An author, defined by his name have several books. A book, defined by its title and publishing year, has one author only. What's this simple database scheme. Please draw it!

Q4 - A user, defined by his email, can read several books. A book (e-book!!) can be read by several user. We also want to keep track of reading dates. Improve your e-library DB scheme with relevant tables and relationships.

Q5 - What's the language to make queries to a database?

Q6 - What's the simple query to get books written before 1985?

Q7 - What's the simple query to get 3 last books written by Jules Verne?

Q8 - What's the purpose of ActiveRecord?

Q9 - What's a migration? How do you run a migration?

Q10 - Complete migrations to create your e-library database.
```ruby class CreateBooks < ActiveRecord::Migration def up create_table :books do |t| end end end ``` ```ruby class CreateAuthors < ActiveRecord::Migration def change end end ``` ```ruby class CreateUsers < ActiveRecord::Migration def change end end ```
```

Q11 - Write a migration to add a column `category` to the books table.

```ruby class AddCategoryToBook < ActiveRecord::Migration def change end end ```

Q12 - Define an ActiveRecord model for each table of your DB. Add the ActiveRecord associations between models.

Q13 - Complete following code using the relevant ActiveRecord methods.

```ruby #1. Add your favorite author to the DB #2. Get all authors #3. Get author with id=8 #4. Get author with name="Jules Verne", store it in a variable: jules #5. Get Jules Verne's books #6. Create a new book "20000 Leagues under the Seas", it's missing in DB. # Store it in a variable: twenty_thousand #7. Add Jules Verne as this book's author #8. Now save this book in the DB! # The end ```

Q14 - Add validations of your choice to the `Author` class. Can we save an object in DB if its validations do not pass?